Kotlin - Ranges

Kotlin **range** is defined by its two endpoint values which are both included in the range. Kotlin ranges are created with **rangeTo()** function, or simply using **downTo** or **(. .)** operators. The main operation on ranges is **contains**, which is usually used in the form of **in** and **!in** operators.

Example

```
1..10  // Range of integers starting from 1 to 10

a..z   // Range of characters starting from a to z

A..Z   // Range of capital characters starting from A to Z
```

Both the ends of the range are always included in the range which means that the 1..4 expression corresponds to the values 1,2,3 and 4.

Creating Ranges using rangeTo()

To create a Kotlin range we call **rangeTo()** function on the range start value and provide the end value as an argument.

Example

```
fun main(args: Array<String>) {
   for ( num in 1.rangeTo(4) ) {
     println(num)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
1
2
3
4
```

Creating Ranges using .. Operator

The **rangeTo()** is often called in its operator form **..**. So the above code can be re-written using **..** operator as follows:

Example

```
fun main(args: Array<String>) {
   for ( num in 1..4 ) {
     println(num)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
1
2
3
4
```

Creating Ranges using downTo() Operator

If we want to define a backward range we can use the **downTo** operator:

Example

```
fun main(args: Array<String>) {
   for ( num in 4 downTo 1 ) {
     println(num)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
4
3
2
1
```

Kotlin step() Function

We can use **step()** function to define the distance between the values of the range. Let's have a look at the following example:

Example

```
fun main(args: Array<String>) {
   for ( num in 1..10 step 2 ) {
```

```
      println(num)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
1
3
5
7
9
```

Kotlin range of Characters

Ranges can be created for characters like we have created them for integer values.

Example

```
fun main(args: Array<String>) {
   for ( ch in 'a'..'d' ) {
      println(ch)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
a
b
c
d
```

Kotlin reversed() Function

The function **reversed()** can be used to reverse the values of a range.

Example

```
fun main(args: Array<String>) {
   for ( num in (1..5).reversed() ) {
      println(num)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
5
4
3
2
1
```

Kotlin until() Function

The function **until()** can be used to create a range but it will skip the last element given.

Example

```
fun main(args: Array<String>) {
   for ( num in 1 until 5 ) {
     println(num)
   }
}
```

When you run the above Kotlin program, it will generate the following output:

```
1
2
3
4
```

The last, first, step Elements

We can use **first**, **last** and **step** properties of a range to find the first, the last value or the step of a range.

Example

```
fun main(args: Array<String>) {
   println((5..10).first)
   println((5..10 step 2).step)
   println((5..10).reversed().last)
}
```

When you run the above Kotlin program, it will generate the following output:

```
5
2
5
```

Filtering Ranges

The **filter()** function will return a list of elements matching a given predicate:

Example

```
fun main(args: Array<String>) {
   val a = 1..10
   val f = a.filter { T -> T % 2 == 0 }

   println(f)
}
```

When you run the above Kotlin program, it will generate the following output:

```
[2, 4, 6, 8, 10]
```

Distinct Values in a Range

The **distinct()** function will return a list of distinct values from a range having repeated values:

Example

```
fun main(args: Array<String>) {
   val a = listOf(1, 1, 2, 4, 4, 6, 10)

   println(a.distinct())
}
```

When you run the above Kotlin program, it will generate the following output:

```
[1, 2, 4, 6, 10]
```

Range Utility Functions

There are many other useful functions we can apply to our range, like **min**, **max**, **sum**, **average**, **count**:

Example

```
fun main(args: Array<String>) {
   val a = 1..10

   println(a.min())
   println(a.max())
   println(a.sum())
   println(a.average())
   println(a.count())
}
```

When you run the above Kotlin program, it will generate the following output:

```
1
10
55
5.5
10
```

Quiz Time (Interview & Exams Preparation)

Q 1 - Which of the following is true about Kotlin Ranges?

A - Kotlin range is a sequence of values defined by a start, an end, and a step.

B - Kotlin range can be created using the rangeTo() and downTo() functions or the .. operator.

C - We can use ranges for any comparable type.

D - All of the above

Q 2 - What will be the output of the following program:

```
fun main(args: Array<String>) {
   val a = 1..20

   println(a.average())
}
```

A - This will print 10.5

B - This will raise just a warning

C - Compilation will stop with error

D - None of the above

Q 2 - What will be the output of the following program:

```
fun main(args: Array<String>) {
   val a = 1..20

   if( 3 in a){
      print( true )
   }else{
      print( false )
   }
}
```

A - true

B - false

C - Compilation will stop with error

D - None of the above